UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

- TRABAJO FIN DE GRADO -

SEGUIMIENTO DE OBJETOS EN TIEMPO REAL

Jorge Sanjuán García
Julio 2014

SEGUIMIENTO DE OBJETOS EN TIEMPO REAL

AUTOR: Jorge Sanjuán García

TUTOR: Rafael Martín Nieto
PONENTE: José M. Martínez Sánchez

Video Processing and Understanding Lab - UAM
Dpto. de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio 2014

Proyecto parcialmente financiado por el gobierno español bajo el proyecto TEC2011-25995 (Event Video)

Resumen y Palabras Clave

Resumen

El objetivo principal de este trabajo es definir un procedimiento general para la integración de algoritmos de vídeo-seguimiento de objetos en la plataforma de análisis distribuido de vídeo DiVA, permitiendo así la ejecución de estos algoritmos de forma *online* y permitiendo la interactividad del usuario durante la ejecución. Gracias a esto y pudiendo usar como referencia los algoritmos ya integrados en la plataforma, futuros algoritmos podrán ser integrados de la misma manera con un esfuerzo mucho menor y pudiendo aprovechar gran parte del trabajo aquí realizado.

Tras la preparación de los algoritmos, se han desarrollado demostradores para cada algoritmo considerado, facilitando la visualización de los resultados y la interacción entre con el usuario gracias a la interfaz gráfica de usuario (GUI).

Adicionalmente, se han introducido mejoras a los algoritmos con los que se ha trabajado. Para el desarrollo de esta etapa de investigación, ha sido necesario realizar un estudio exhaustivo sobre la implementación y la base teórica de éstos. En concreto, la mayor parte de los esfuerzos se han centrado en el segundo algoritmo integrado, PKLTF. Tras las mejoras implementadas, se han estudiado objetivamente los resultados del algoritmo utilizando un marco de evaluación previamente definido, consiguiendo demostrar que las mejoras introducidas mejoran el rendimiento del *tracker*.

Palabras Clave

Seguimiento de objetos, procesamiento de video, sistema mono-cámara, sistema mono-objetivo, interfaz gráfica de usuario, evaluación, demostradores.

Abstract and Keywords

Abstract

The main goal of this work is to define a general method for integration of several video object tracking algorithms into the Distributed Video Analysis platform DiVA, allowing the algorithms to process video sequences online and interactivity between the user and the algorithm during the execution. Thanks to this, by using as a reference the algorithms integrated in this work, future integrations of new algorithms can be developed in the same way with much less effort, taking advantage of the developed work.

After preparing the algorithms, demonstrators for each integrated algorithm have been developed in order to ease verification of the results and interactivity between the user and the algorithm trough the graphic user interface (GUI) created.

Furthermore, improvements have been developed to the algorithms treated in this work. To carry out this investigation task, a comprehensive study on the implementation and theoretical basis has been made to each algorithm. Specifically, most of the efforts has been focalized in the second integrated algorithm, PKLTF. After the implementation of the improvements, both the results and processing time have been studied objectively using the previously defined evaluation framework, verifying that, with the improvements, the algorithm offers better results.

Keywords

Object tracking, video-processing, single-camera system, mono-target system, graphic user interface, evaluation, demonstrators.

Agradecimientos

Quiero agradecer este trabajo al colectivo de profesores de la escuela por haber hecho este plan de estudios en el que he tenido la oportunidad de formarme y obtener una gran cantidad de conocimientos con gran profundidad y extensión. Agradecer también este trabajo a mi tutor del trabajo, Rafa; mi ponente, Chema; a todos los compañeros del laboratorio y, en general, a todo el personal del grupo VPU.

Jorge Sanjuán García
Julio 2014

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

Glosario de acrónimos

- **TM:** Template Matching.
- **MS:** Mean-Shift.
- **PFC:** Particle Filter-based Color.
- **LK:** Lucas-Kanade.
- **IVT:** Incremental learning for Visual Tracking.
- **TLD:** Tracking Learning Detection.
- **CBWH:** Corrected Background-Weighted Histogram.
- **SOAMST:** Scale and Orientation Adaptive Mean-Shift Tracking.
- **PKLTF:** Point-based Kanade Lukas Tomasi color-Filter tracker.
- **DiVA:** Distributed Video Analysis framework.
- **IDE:** Integrated Development Enviroment.
- **SOVTds:** Single Object Video Tracking dataset.
- **VOT:** Video Object Tracking.
- **SFDA:** Sequence Frame Detection Acuracy
- **FPS:** Frames per Second. Fotogramas Por Segundo
- **GUI:** Graphic User Interface

1 Introducción

1.1 Motivación

Desde la aparición del video digital, la investigación en el procesado de vídeo está en continuo crecimiento. Las estrategias y los objetivos que se persiguen detrás de este campo son muy diversos y, en particular, debido a la gran demanda existente, los sistemas de vídeo vigilancia ocupan gran parte de la tarea de investigación del procesado de vídeo.

En este contexto, el vídeo-seguimiento de objetos es una de las ramas más importantes en las tareas de procesado de vídeo. Conocer la localización de un objeto en una secuencia de vídeo puede dar mucha información relevante sobre lo que está ocurriendo en ella. Por esto, el video-seguimiento de objetos no es un campo únicamente útil en los sistemas de vídeo-vigilancia si no que, además, puede ser extendido a muchas otras áreas como la robótica, control de cámaras móviles, aplicaciones multimedia, post-producción digital, compresión de formatos de vídeo, etc.

Por otro lado, los avances en computación e implementación de algoritmos han hecho posible que estos sistemas de vídeo-seguimiento de objetos puedan trabajar a tiempo real y por ello, llevar a cabo la creación de aplicaciones que puedan mostrar en tiempo real el funcionamiento de éstos puede incrementar el interés tanto de desarrolladores, consumidores, estudiantes, etc, ayudando así a la rápida evolución de estos sistemas.

Además, en sistemas de estas características, los resultados que se obtienen son, en gran parte, un resultado subjetivo en el que la tarea de conocer si el algoritmo está obteniendo unos resultados acertados es difícil y no existen unos métodos estandarizados para su evaluación, conllevando a que cada desarrollador emplee sus propias métricas de medida.

Parte de este trabajo ha sido presentado en el Workshop:"Strategies for object Segmentation, Detection and Tracking in Complex Enviroments for Event Detection in Video Surviellance and Monitoring", TEC2011-25995 EventVideo (2012-2014).

1.2 Objetivos

Los objetivos principales de este trabajo son: integrar algoritmos de vídeo-seguimiento de objetos en la plataforma de vídeo-vigilancia DiVA definiendo una metodología genérica que facilite la integración de futuros algoritmos, el desarrollo de interfaces gráficas para acercar esta tecnología a cualquier tipo de usuario, realizar mejoras en un algoritmo y evaluación de resultados de estas mejoras.

A continuación se describen las tareas realizadas para cumplir los objetivos:

1. Estudio del estado del arte.
    - Conocer en qué consiste el vídeo-seguimiento de objetos y cuáles son sus principales aplicaciones y limitaciones.

- Estudiar las principales técnicas utilizadas para esta tarea tanto a nivel teórico como a nivel de implementación.

2. Aprendizaje del entorno de trabajo.
   - Introducirse en los conceptos básicos de la programación orientada a objetos para el desarrollo de aplicaciones de procesado en C++.
   - Estudio de la librería de procesado de imagen OpenCV.
   - Estudio de la librería QT de creación de aplicaciones con interfaz gráfica.
   - Estudio de la plataforma de vídeo-vigilancia DiVA.
   - Familiarización con los sistemas de evaluación de algoritmos existentes en el VPULab.
3. Integración de algoritmos.
   - Estudio de los algoritmos disponibles.
   - Adaptación de los mismos al formato de trabajo de la plataforma DiVA.
4. Realización de interfaces gráficas.
   - Implementación de un método de entrada generalizado para algoritmos de vídeo-seguimiento.
   - Realización de las aplicaciones.
5. Implementación y evaluación de mejoras en algoritmos de vídeo-seguimiento.
   - Problemas observados.
   - Implementación de las mejoras.
   - Evaluación tras las mejoras.

1.3 Organización de la memoria

Una vez definidos los objetivos, la estructuración con la que se presentan es la siguiente:

- **Capítulo 1:** Introducción motivación y objetivos del proyecto.
- **Capítulo 2:** Definición de video seguimiento, estudio del estado del arte de los diferentes algoritmos utilizados en este trabajo, introducción de la plataforma de desarrollo DiVA y la librería de desarrollo de interfaces gráficas QT.
- **Capítulo 3:** Desarrollo de la integración de algoritmos en la plataforma DiVA.
- **Capítulo 4:** Desarrollo de interfaces gráficas que demuestran el funcionamiento de los algoritmos.
- **Capítulo 5:** Estudio del sistema y repositorios necesarios para la evaluación de algoritmos de vídeo-seguimiento.
- **Capítulo 6:** Implementación de mejoras al algoritmo PKLTF y su posterior estudio objetivo de resultados y tiempos de procesado.
- **Capítulo 7:** Conclusiones y trabajo futuro.

2 Estado del arte

En esta sección se presenta una visión general del trabajo previo que se ha realizado en el campo del estudio presentado en este trabajo fin de grado. En las siguientes subsecciones se presenta el video seguimiento de objetos (sección 2.1), los algoritmos del estado del arte que han sido relevantes en la realización del trabajo (sección 2.2), las herramientas de integración de algoritmos de vídeo vigilancia en la plataforma distribuida DiVA (sección 2.3) y las herramientas utilizadas para el desarrollo de aplicaciones finales para usuarios (sección 2.4).

2.1 Vídeo-seguimiento de objetos

El video seguimiento de objetos es una de las ramas principales en la visión artificial y consiste en una técnica de procesado de video que trata de estimar la posición de uno o varios objetos determinados en cada uno de los *frames* de una secuencia de video y que, ésta, a su vez, puede estar capturada por múltiples cámaras.

Un objeto podría definirse como cualquier elemento que aparece en una imagen y se considere de interés. De esta manera, un algoritmo de seguimiento de objetos necesita una entrada que defina cuál es el objeto que, típicamente, queda definido como las coordenadas de los píxeles que forman un polígono de tal manera que el objeto quede en su interior. La salida del algoritmo serán entonces las coordenadas del polígono en la nueva posición en la que se estima que se encuentra el objeto. Sin embargo, si algoritmo es capaz de adaptarse a los cambios de escala y/o orientación del objeto, el polígono a la salida puede verse modificado.

Respecto a la arquitectura de un sistema de estas características, queda definida con cinco bloques fundamentales [1]:

- **Extracción de características:** trata de extraer información relevante sobre el área de la imagen en la que se encuentra el objeto. Ésta puede estar basada en color, movimiento, etc.
- **Representación del objetivo:** para definir la apariencia del objeto seguido. Las características consideradas para diferenciarlo del resto de objetos de la secuencia.
- **Localización:** Permite estimar la propagación del objeto a lo largo del tiempo. Esta localización será estimada a partir de la información extraída en los bloques anteriores.
- **Gestión del seguimiento:** se debe seguir una estrategia en el seguimiento del objeto cuando éste aparece o desaparece parcial o totalmente del plano de imagen.
- **Extracción de metadatos:** para ser usados por la aplicación.

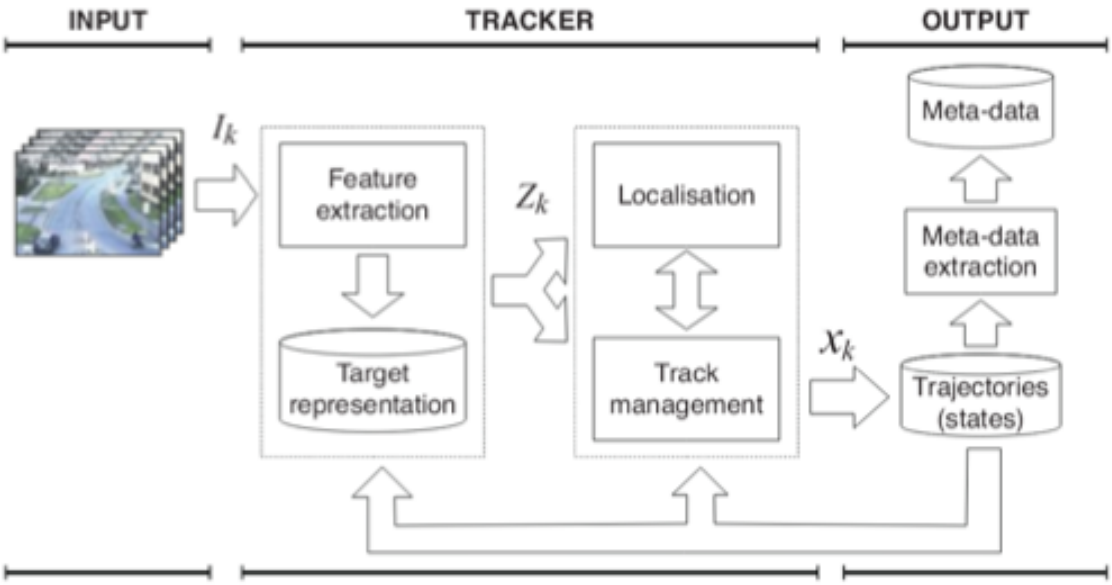

Figura 2-1: Arquitectura de un sistema de video-seguimiento de objetos. Imagen extraída de [1].

La extracción de características es fundamental en el seguimiento de objetos. Estás definen cuál es el modelo con el que el objeto es seguido y, por lo tanto, deben definir únicamente el objeto a seguir, aislándolo de la información del fondo o de otros objetos similares que puedan aparecer en la secuencia. Además, debido a los diversos problemas que pueden surgir, no es trivial la elección de un método de extracción de características ya que cada uno podrá adaptarse a ciertos problemas en función de la situación, siendo muy difícil que se adapte a la totalidad de los problemas. La extracción de características puede agruparse en tres niveles [1]:

1. **Bajo nivel**: ej. color, gradientes, movimiento.
2. **Nivel Medio**: ej. esquinas, regiones.
3. **Alto nivel:** ej. segmentación frente/fondo, modelos predefinidos.

La tarea de seguimiento es complicada debida a que aparecen múltiples problemas. Centrarse en ciertos problemas pueda afectar negativamente a la solución de otros. Los problemas más comunes a los que un seguidor de objetos tiene que hacer frente son los siguientes:

- **Movimiento rápido o complejo:** ante este problema, el sistema puede perder el objeto momentáneamente o, incluso por completo si el área de búsqueda no es lo suficientemente grande.
- **Cambios de iluminación:** Los posibles cambios de iluminación pueden cambiar el nivel de intensidad de los pixeles con los que el objeto es observado, alejando su apariencia del modelo aprendido inicialmente.
- **Ruido:** se muestra como variaciones aleatorias al nivel de los pixeles de la imagen y pueden interferir en los resultados del algoritmo.
- **Oclusiones:** esta situación se produce cuando un objeto se interpone entre la cámara y el objetivo. Puede ser parcial o total.

- **Cambios de escala:** generalmente ocurren cuando la distancia entre el objeto y la cámara se ve afectada.
- **Objetos similares:** la similitud entre el objetivo y otros objetos similares puede dar lugar a confusiones y resultados erróneos a la salida del seguidor de objetos.

2.2 Algoritmos de video-seguimiento

En este apartado se describen algunas de las técnicas más utilizadas para el video-seguimiento de objetos de carácter mono-objetivo. Esto es, sólo se pretende localizar un objeto en cada *frame* de la secuencia de video.

2.2.1 Template Matching (TM)

Esta técnica de video-seguimiento de objetos [2] es una de las más simples usadas en el procesado de video. La idea principal es la de seleccionar una imagen, que puede corresponder a una región de interés de un *frame* de la secuencia, y ésta se utilizará como plantilla. Una vez seleccionada la plantilla, se compararán las intensidades de los pixeles con las posibles posiciones que pueda tomar en el siguiente *frame* y, finalmente, la localización candidata será aquella que proporcione una mayor confianza según el sistema de medida escogido. Típicamente: medidas de comparación de sumas, correlación o convolución.

2.2.2 Mean-Shift (MS)

En esta técnica de seguimiento de objetos [3] la idea principal es la de construir un histograma del objeto a seguir y de los posibles candidatos en el siguiente *frame*. De esta manera, se tiene información del color de la región de interés, no siendo la información a nivel de pixel. Además, la región de interés se determina generalmente de manera elíptica para así poder asignar pesos, de tal manera que, los pixeles correspondientes al centro del objeto sean más relevantes para la realización del histograma que los pixeles más alejados del centro, los cuales se asume que pueden ser menos relevantes al estar más cercanos a los bordes del objeto.

2.2.3 Particle Filter-based Colour tracking (PFC)

Este algoritmo de seguimiento de objetos [4] presenta una técnica en la que la búsqueda de la posición del objeto es similar a la empleada en el método Mean-Shift, ya que también se basa en color, pero las posiciones candidatas para localizar el objeto en un *frame* determinado se modelan como una media ponderada de diferentes posiciones candidatas. Cada una de éstas se considera una partícula (con un peso asociado a cada una) que, de manera iterativa, cambiarán su posición y tamaño de manera pseudo-aleatoria hasta que el algoritmo converge en una posición determinada.

2.2.4 Lukas-Kanade tracking (LK)

Este algoritmo de seguimiento de objetos [5] presenta una técnica similar a Template Matching, pero con la particularidad de que permite pequeños cambios afines en la imagen plantilla (translaciones, rotaciones, cambios de escala, etc.). Por lo tanto, se estima la localización del objeto mediante la búsqueda de los parámetros que definen una cierta transformación afín que ajusta la imagen actual a la plantilla o sub-imagen que define el objeto.

2.2.5 Incremental learning for robust Visual Tracking (IVT)

Este algoritmo de seguimiento de objetos [6] tiene la característica de aprender y adaptarse a cambios en la apariencia del objeto. Por lo tanto, mientras que la mayoría de los algoritmos existentes no tienen en cuenta que la apariencia del objeto o el ambiente que lo rodea pueden cambiar, éste se adapta para obtener un modelo de objeto más actualizado a las condiciones de iluminación o apariencia en un determinado momento, facilitando así la labor de búsqueda. Esta adaptación es posible debido a la linealidad que se asume en los cambios que puedan aparecer entre *frames* consecutivos, permitiendo así la creación de una auto-base de aprendizaje que, mientras los cambios no sean bruscos, será capaz de aprender con certeza la evolución del objeto a lo largo de la secuencia.

2.2.6 Tracking Learning Detection tracking (TLD)

Este algoritmo de seguimiento de objetos [7] con capacidad de seguimiento a largo plazo, combina un bloque de seguimiento de movimiento con un bloque de detección. El bloque de seguimiento estima la localización del objeto a seguir a partir de información de movimiento entre *frames* consecutivos. En el bloque de detección se realiza una búsqueda exhaustiva en toda la imagen con la intención de localizar al objeto a partir de información sobre el objeto que haya sido extraída en *frames* anteriores. Por último, mediante un componente de aprendizaje, con el objetivo de que el sistema aprenda cuando se han producido errores en la detección (falsos positivos o falsos negativos), se comparan los resultados obtenidos en el seguimiento y en la detección.

2.2.7 Corrected Background-Weighted Histogram tracker (CBWH)

Este algoritmo de seguimiento de objetos [8] presenta una modificación sobre el algoritmo Mean-Shift (descrito en 2.2.2) con la idea de reducir la información de color del fondo que puede aparecer en el histograma en la inicialización del objeto. De esta manera, el modelo del objeto a localizar queda menos distorsionado por la información del fondo. El algoritmo sólo aplica la corrección de histograma en la inicialización del objeto de tal manera que la información de color sea únicamente la del objeto a seguir y no pueda compararse con la del fondo en *frames* posteriores.

2.2.8 Scale and Orientation Adaptative Mean-Shift Tracking (SOAMST)

Este algoritmo de seguimiento de objetos [9] es una modificación del algoritmo original de Mean-Shift descrito anteriormente. Por lo tanto, es un algoritmo basado en la información de color del objeto, pero éste trata de actualizar los cambios de orientación y escala del objetivo durante el proceso de búsqueda. Esto es posible debido a que la imagen de pesos generada por el algoritmo Mean-Shift no es sólo usada para estimar la localización del objeto mediante la correspondencia de los valores del histograma de color, si no que, además, esta información es también usada para estimar los cambios en el área y la orientación en los *frames* consecutivos.

2.2.9 PKLT Filter Tracker (PKLTF)

PKLTF (Point-based Kanade Lukas Tomasi color-Filter Tracker) [10] es un algoritmo de video-seguimiento de objetos desarrollado por Antonio González Huete en el VPULab como parte de su trabajo final de Máster en el año 2013.

Éste es un algoritmo en el que el objeto se modela mediante la fusión de distintos tipos de características: puntos característicos y color. Estás características serán modeladas como partículas, que podrán ser filtradas en caso de que éstas se consideren degeneradas o, de otra manera, su característica asociada ya no sea relevante.

Estas partículas definen puntos característicos de la imagen. En particular, se consideran características las esquinas y bordes, y son detectadas mediante el detector de esquinas propuesto por Shi-Tomasi [11], en el que se hace uso del detector de bordes Harris y Stephens [12]. Una vez obtenidos los puntos característicos, se obtendrán las partículas KLT utilizadas en este algoritmo. Estas partículas provienen del trabajo "KLT feature tracker", presentado por Shi-Tomasi en [11], y son capaces de describir vectores de desplazamiento con respecto al tiempo.

Una vez detectadas las partículas, se inicializan dos descriptores a partir de éstas: uno de movimiento y otro de color. Hecho esto, si el descriptor en los *frames* posteriores diverge con respecto a los descriptores originales, se considera una degeneración y las partículas que no son relevantes serán filtradas mientras que a las partículas relevantes se las asignaran nuevos pesos.

Este algoritmo posee además de una segunda etapa de refinamiento de la posición mediante ascenso por gradiente para llevar los resultados al punto más óptimo con respecto al modelo de objeto. En esta optimización, se hace uso de una etapa de estimación de la localización con información de color mediante la implementación descrita en [13] del método Mean-Shift (definido en la sección 2.2.2).

2.3 Distributed Video Analysis Framework (DiVA)

En esta sección se describe el entorno de trabajo en el que se han integrado algoritmos de seguimiento de objetos.

La plataforma DiVA ha sido desarrollada por el VPULab en la Escuela Politécnica Superior de la Universidad Autónoma de Madrid [14]. Esta plataforma permite realizar el flujo de trabajo completo necesario para el procesamiento de secuencias de video a tiempo real, de manera modular y en la red. Es decir, establece un sistema distribuido para la adquisición de múltiples secuencias de video (de cámaras o de archivos), comunicación y aplicación de diferentes algoritmos de procesamiento en serie o paralelo y visualización parcial de los resultados en tiempo real.

Los criterios con los que la plataforma fue creada son:

- Escalabilidad: se deben poder añadir módulos adicionales para el procesado de algoritmos y ser compatible con distintas fuentes de video (FireWire, Ip, etc) o protocolos de codificación (MPEG2, h.264, etc).
- Eficiencia: no puede alterar significativamente los tiempos de procesado del algoritmo.
- General: debe estar construido con el menor número de paquetes de herramientas y operaciones adicionales.
- Tolerancia ante errores: debe incluir mecanismos de detección y corrección de errores.

Para la integración de algoritmos en la plataforma DiVA en el VPULab se describe la metodología a seguir [15]. Ésta consta de tres etapas o niveles principales:

1. **Preparación del algoritmo:** en esta etapa se debe crear una clase en C++ en la que, siguiendo una serie de pautas en las que se detallan cómo deben crearse lo métodos de la clase de integración, el algoritmo quedará listo para ser encapsulado en la siguiente fase. Esta preparación puede ser realizada desde cero (implementando el algoritmo en C++) o a partir de código que provenga de otro lenguaje de programación o incluso del propio C++.
2. **Encapsulación del algoritmo:** en el encapsulador DiVAAlgorithm se integra el algoritmo previamente preparado con el formato predefinido en el nivel anterior. En esta fase, el algoritmo ya es capaz de funcionar de manera que la imágenes de entrada provengan de la red en lugar de un fichero.
3. **Desarrollo de la aplicación y de la interfaz de usuario:** con el objetivo de acercar el algoritmo al usuario final, se crea una aplicación en la que un usuario no necesariamente experto puede hacer funcionar y/o configurar el algoritmo y ver los resultados finales o parciales. Esta interfaz además permite a desarrolladores de algoritmos realizar pruebas de forma sencilla y directa.

2.4 Plataforma QT

QT es una librería multiplataforma [16], desarrollada como software libre y de código abierto, con herramientas diseñadas para llevar a cabo la creación de aplicaciones y de interfaces gráficas de escritorio o dispositivos móviles. De esta manera y con gran eficiencia, QT facilita la estandarización en la realización de interfaces gráficas para varias plataformas con un único código base.

Con las herramientas que la biblioteca proporciona es posible desarrollar una gran variedad de interfaces gráficas y de gran complejidad. De hecho, un gran número de aplicaciones comerciales punteras han sido desarrolladas mediante las herramientas que esta librería ofrece.

Además, al tratarse de una herramienta para la realización de interfaces gráficas, es conveniente la utilización de programas externos que permitan la programación visual, facilitando así la programación de la distribución de los componentes que forman una interfaz gráfica. Sin embargo, la librería con su herramienta para la programación visual, puede ser utilizada como plugg-in de algunos IDE (Integrated Development Enviroment) como, por ejemplo, Microsoft Visual Studio, el cual ha sido el entorno de desarrollo principal para la programación en C++.

Las interfaces gráficas realizadas en este trabajo, correspondientes a la implementación del Nivel 3 descrito en el apartado anterior, se han implementado utilizando la librería QT.

3 Desarrollo

3.1 Integración de algoritmos

En este apartado se describe en qué consiste la integración de algoritmos en la plataforma de desarrollo DiVA del VPULab y, posteriormente la realización de una interfaz gráfica o demostrador de los algoritmos. Este bloque engloba gran parte de la realización de este trabajo en el que se han seleccionado los algoritmos que podrían ser utilizados por sus características, se ha investigado la existencia de implementaciones de los mismos ya sea de forma de código libre o por implementaciones ya realizadas por el VPULab y, por último, se ha llevado a cabo la implementación de la integración y desarrollo de una interfaz de usuario de dos *trackers*; Template Matching (descrito en la sección 2.2.1) y PKLT Filter Tracker (descrito en la sección 2.2.9).

En la selección del sistema de video-seguimiento se plantean algunas restricciones. En primer lugar, se plantea el primer objetivo: el *tracker* debe ser un sistema que sea capaz de funcionar en tiempo real. Debido a que la cantidad de datos que un sistema de este tipo maneja es muy elevada, los tiempos de búsqueda del objeto pueden ser muy largos, sobre todo si se emplean técnicas exhaustivas o de extracción de características complejas. Es por esto que el nivel de exigencia de algunos *trackers* queda reducido cuando el sistema detecta que su estimación es correcta, pero, sin embargo, si el sistema detecta que ha perdido el objeto, el nivel de exigencia en la búsqueda suele ser más exhaustivo y computacionalmente más costoso, aumentando el tiempo de cómputo y perdiendo el factor de tiempo real. Por eso, si esta situación se da, el sistema debe ser capaz de recuperar el objeto en el menor tiempo posible. En segundo lugar, el algoritmo debe funcionar a largo plazo. Esto es, debe ser capaz de seguir al objeto objetivo durante largos periodos de tiempo y no perderlo en la medida de lo posible y, si lo perdiese, ser capaz de recuperarlo y, como se ha comentado anteriormente, en un tiempo razonable. Esto es un punto muy importante en la selección del algoritmo ya que no todos cumplen esta restricción y limitan la búsqueda del objeto a unos segundos o algunos minutos.

Se decidió, con motivo de crear una primera aplicación en la que se estableciese una metodología para integraciones posteriores, la integración de una implementación de Template Matching disponible en el VPULab. Esta implementación realizaba de forma exhaustiva la búsqueda del objeto en todo momento y, tratando de minimizar los tiempos de computo, suponiendo que el movimiento del objeto entre *frames* consecutivos no necesita la búsqueda exhaustiva para la localización del mismo, se implementa una reducción del área de búsqueda del algoritmo, consiguiendo así una menor cantidad de datos a procesar por cada frame. Sin embargo, se mantiene la posibilidad de aumentar el área de búsqueda del objeto en el caso de que se considere perdido el objeto según una medida de confianza.

Por otro lado, con intención de realizar dos aplicaciones completas; la anteriormente descrita y otra que ofrezca unos mejores resultados que Template Matching, se llevó a cabo la integración de un algoritmo más complejo.

Previamente a la implementación que finalmente se realizó, se planificó integrar una implementación en software libre del *tracker* TLD (descrito en el apartado 2.2.6). Esta implementación por Georg Nebehay, es una migración del código originalmente escrito en Matlab por Zdenek Kalal a C++. Se denomina OpenTLD y está disponible en el siguiente sitio web [17].

Finalmente, la integración que se realizó fue la del algoritmo PKLT Filter Tracker debido a los buenos resultados que este ofrece, la gran robustez y la capacidad de funcionamiento a largo plazo pues realiza una actualización constante del modelo con el que busca al objeto. Además este algoritmo es capaz de funcionar en tiempo real, a diferencia del TLD. Este es el principal motivo por el que se decidió integrar el algoritmo PKLTF y descartar la integración del tracker TLD.

Por lo tanto, como se ha dicho anteriormente, se realizó la integración completa de dos algoritmos: Template Matching y PKLT Filter Tracker. Esta implementación queda dividida en tres partes, detalladas a continuación, tal y como se describe en la sección 2.3.

3.1.1 Nivel 1

En este nivel, el objetivo principal es la preparación del algoritmo que se desea integrar en la plataforma DiVA. Se trata de una etapa en la que se define una clase en C++ de tal manera que se acoja a las pautas de integración de la plataforma.

En este nivel, las aplicaciones desarrolladas reciben como entrada el directorio y fichero que contiene una secuencia de video y, en su caso (PKLTF), un fichero de configuración. En la secuencia de video entrante se debe indicar en primer lugar la localización del objeto a seguir mediante la lectura de un fichero de *ground-truth.* Una vez seleccionada la región en la que se encuentra el objeto, el algoritmo queda completamente inicializado y, por lo tanto, se podrá llamar en un bucle al método de procesado de imágenes, el cual recibe como entrada cada uno de los *frames* de la secuencia de video.

La salida principal de ambos algoritmos es la misma secuencia de video, pero con la representación de un cuadrilátero o *bounding box* que indica la localización estimada del objeto para cada *frame.* Para la integración de PKLTF, también se representan los puntos característicos que el algoritmo tiene en cuenta para la estimación de la localización del objeto. Además, se implementa un método de captura de resultados en el que se crea una secuencia de video con la representación de los resultados.

Se ha implementado un sistema de recogida automática de datos de salida para su futura evaluación objetiva. Este sistema es capaz de extraer las localizaciones iniciales desde ficheros de *ground-truth*, procesar todos los ficheros de vídeo contenidos en un directorio y anotar la localización estimada del objeto para cada *frame* en un archivo de texto plano. Además, en este nivel se implementan las mejoras realizadas al algoritmo original PKLTF, que se describen en la sección 6 de este trabajo.

3.1.2 Nivel 2

En este nivel, con el algoritmo ya preparado en el Nivel 1, se integra la clase creada en la plataforma DiVA. Para ello, se crea una nueva clase que hereda de la clase principal de procesado de DiVA y hace uso de los métodos de la clase implementada en el Nivel 1 mediante la declaración de un objeto de ésta.

La entrada de video de esta aplicación es ahora un servidor de *frames* disponible en la plataforma. Éste puede transmitir vídeos desde archivo o vídeos capturados por una cámara en tiempo real.

La salida del algoritmo en este nivel es similar a la del nivel anterior, pero, al tratarse de una aplicación enfocada a la recepción de *frames* en tiempo real capturados por una cámara, la declaración del objeto no es posible realizarla por entrada de código o por fichero de *ground-truth* ya que no se puede estimar la localización del mismo a priori. Por lo tanto, es necesario un método de entrada que permita seleccionar el objeto en la imagen. Este método se implementa en el Nivel 3 para su utilización en la interfaz gráfica.

3.1.3 Nivel 3

En este nivel se crea una aplicación (ver sección 4) en la que la configuración y visualización de los resultados se realizan de forma más rápida y sencilla. Se ha desarrollado mediante la librería para la realización de interfaces gráficas QT, descrita en la sección 2.4 de este trabajo.

La fuente de vídeo para esta aplicación es la misma que la del nivel anterior: un servidor de *frames*. Sin embargo, para este nivel, la salida o representación en la interfaz gráfica puede ser: el video sin procesar, sin la inicialización de parámetros ni la localización del objeto o, por el contrario, se podrá representar en las interfaces implementadas los resultados del algoritmo tras su procesado. En cuanto a la selección del objeto, éste se seleccionará con el ratón del ordenador, dibujando un rectángulo donde se encuentre el objeto.

4 Demostradores

En este apartado se describen los demostradores o interfaces gráficas de usuario que se han creado para la representación de los dos algoritmos integrados en la plataforma DiVA.

La motivación por la cual se han creado interfaces gráficas o demostradores de los sistemas de video-seguimiento de objetos con los que se ha trabajado en este trabajo, es la de acercar al usuario no necesariamente experto esta tecnología y, por consiguiente, conseguir un manejo más sencillo de los algoritmos o atraer usuarios desde un punto de vista divulgativo y/o comercial. En este sentido, se pretende que usuarios que no pertenezcan al colectivo de individuos dedicados al procesado de video, ni tampoco a los sistemas de software avanzados, también puedan interactuar con los sistemas desarrollados en este trabajo. Además, para el usuario experto, las aplicaciones facilitan el acceso, la configuración y visualización de estos algoritmos.

En particular, un sistema de video-seguimiento de objetos necesita un método de entrada que, si este no proviene de la salida de algún sistema de procesado de video anterior, debe ser inicializado de manera supervisada. Es decir, introducida manualmente por el usuario.

Con este inconveniente, es necesario que el objeto se indique de manera supervisada aun si el usuario de la aplicación no está especializado o experimentado en el uso de este tipo de software.

4.1 Características comunes

En este trabajo se han realizado dos demostradores; correspondientes al algoritmo de Template Matching y al de PKLT Filter Tracker y, aunque los algoritmos tengan funcionamientos muy distintos, la forma en la que un usuario debe manejar dos sistemas del mismo tipo (video-seguimiento de objetos) debe ser similar.

Bajo este criterio, el punto que define la interactividad principal del usuario con el sistema es el de la entrada del algoritmo. En estos sistemas, la entrada del algoritmo se define como una sub-imagen correspondiente a un *frame* de la secuencia de video y declarada como las coordenadas de la imagen que definen un cuadrilátero encasillando al objeto a seguir.

En cuanto a la definición de la región de interés que encuadra el objeto de manera manual, se pueden ver dos interpretaciones:

- **Inicialización con un *frame* concreto:** en esta interpretación es necesario la representación de un *frame* estático. Éste debe quedar en pantalla continuamente hasta que el usuario seleccione el objeto a seguir.
- **Inicialización en tiempo real:** en esta interpretación el objeto se selecciona en tiempo real. Esto es, se selecciona el objeto en movimiento a medida que la secuencia de video es representada en la interfaz.

Con estas consideraciones, se decidió que el sistema más intuitivo y representativo del algoritmo sería la segunda opción, que permitía la interacción continua por el usuario sin necesidad de pausar la secuencia de video. Respecto a la primera opción, no se considero adecuada la necesidad de detener el video ya que, una vez detenida la secuencia y seleccionado el objeto en una localización determinada, la localización del objeto podría haber cambiado notablemente debido a la llegada constante de *frames*. De todos modos, este es un método de representación en el que el objetivo es mostrar el funcionamiento del algoritmo de manera subjetiva.

La forma más intuitiva de seleccionar la localización del objeto a tiempo real es mediante la entrada de ratón. Este método permite dibujar un recuadro en el que se encuentra el objeto en la pantalla que está representando la secuencia de video, pero, al tratarse de una aplicación en tiempo real, aparecen ciertos problemas de robustez en la inicialización del algoritmo mientras el recuadro es dibujado. Por ello, se han tenido en cuenta en la implementación de este sistema los posibles problemas que pueden hacer que la aplicación deje de funcionar debido a errores en la ejecución durante la selección del recuadro. Estos problemas son:

- **Sincronización con la inicialización:** en este sentido, el algoritmo debe comenzar a procesar y crear el modelo de objeto únicamente si el recuadro está completo y cumple las restricciones de tamaño necesarias. Además, el sistema ofrece robustez ante la re-inicialización con otro objeto, o rectificación del mismo, sin necesidad de detener el algoritmo.
- **Robustez ante la selección en distintas direcciones:** independientemente del sistema de coordenadas que el usuario considere en el proceso de encuadre del objeto, este será transformado y trabajará con valores absolutos para evitar valores negativos.
- **Bordes de la imagen:** si el usuario encuadra el objeto y excede ligeramente los bordes de la imagen, independientemente del borde, el recuadro se corregirá acotándolo por el borde de la imagen.
- **Limitaciones en el tamaño del recuadro:** el tamaño mínimo que puede definir un objeto es de al menos dos pixeles de alto y ancho. El tamaño máximo no debe superar el 60% del ancho o alto del *frame* ya que no tiene sentido la búsqueda de objetos que abarquen tanta superficie de la imagen.

Con estas restricciones se consigue un sistema robusto en la entrada del algoritmo, eliminando así la posible aparición de fallos de programación o *bugs* y permitiendo que cualquier usuario pueda utilizarlo con distintos criterios en la selección de las coordenadas de declaración del objeto.

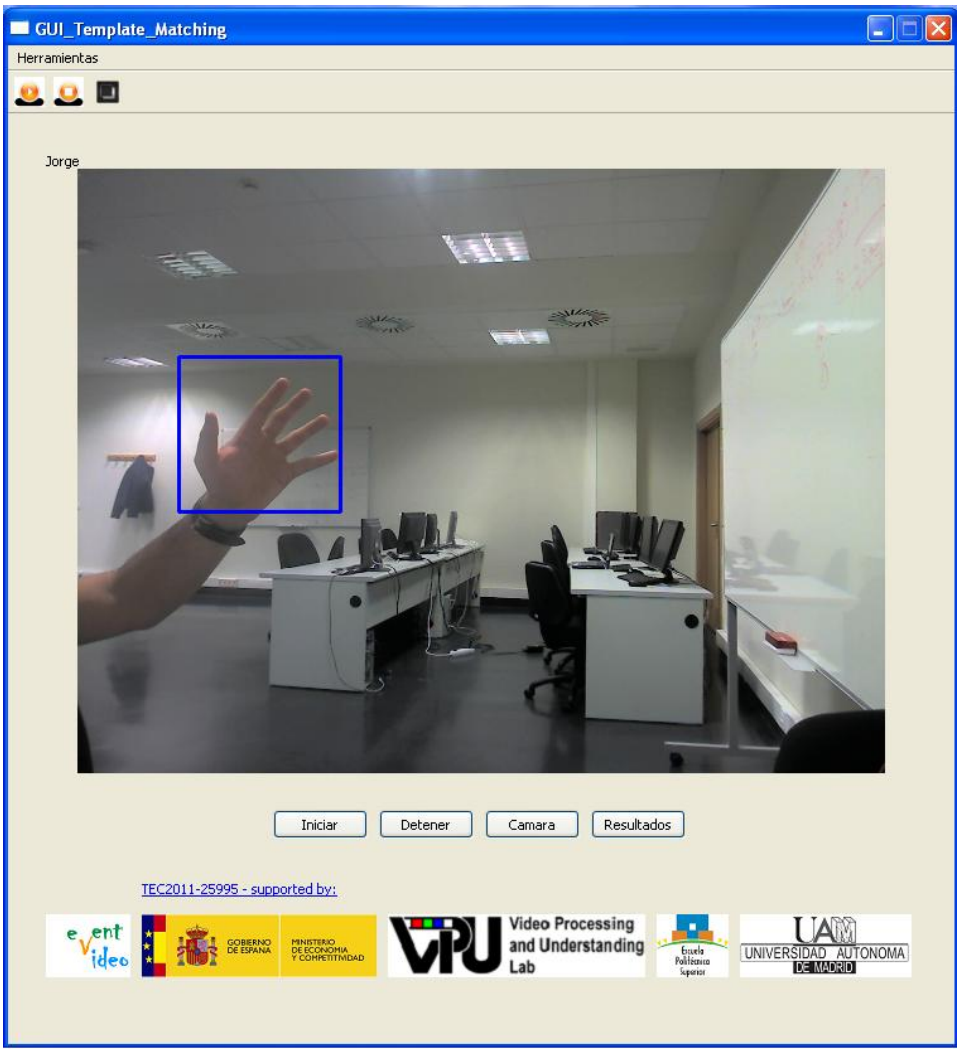

Figura 4-1: Ventana principal de la aplicación Template Matching en funcionamiento.

Selección del servidor de vídeos:

Al tratarse de un sistema distribuido, los vídeos de entrada llegarán desde la red mediante un servidor de *frames*. Para realizar esta conexión cliente-servidor desde la interfaz gráfica, se utiliza una ventana de configuración de cámaras ya disponible en el VPULab [15]. Con ésta, es posible seleccionar entre distintos servidores de *frames* desde un fichero de configuración y, además, se pueden añadir nuevos servidores o establecer uno por defecto desde la aplicación. Para hacer esta venta aparezca, habrá que presionar el botón "cámara" en la ventana principal.

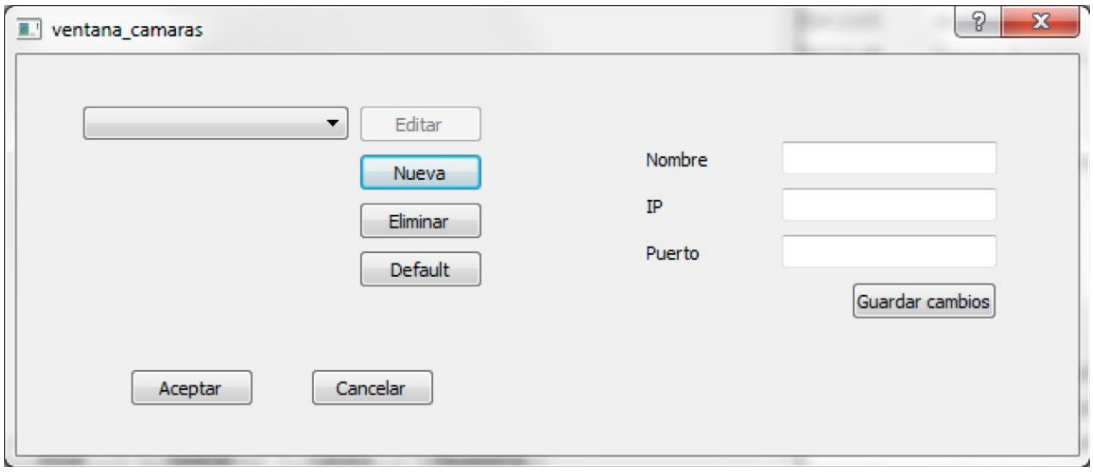

Figura 4-2: Ventana de configuración del servidor de frames.

Los pasos a seguir para hacer funcionar la aplicación son los siguientes:

- Selección del servidor de *frames.* Para ello, se debe presionar el botón "cámara".
- Presionar el botón "Iniciar". A partir de entonces, se mostrará en la aplicación la secuencia de video sin aplicar ningún procesado.
- Seleccionar el objeto el cuál se pretende seguir su movimiento. Para ello, se debe dibujar un rectángulo con el ratón mientras se mantiene presionado el botón izquierdo. Una vez soltado el botón, el algoritmo empezará a realizar la búsqueda. Si se desea modificar, se puede volver al punto 2. presionando en el botón derecho del ratón sobre la imagen o se puede re-dibujar el rectángulo.
- Ajuste de parámetros, visualización de resultados intermedios (*Template Matching*) y visualización de puntos característicos (PKLTF).
- Detener el algoritmo.

4.2 Template Matching

Esta aplicación con interfaz gráfica tiene como objetivo establecer las pautas a seguir en la metodología de creación de aplicaciones para el seguimiento de objetos. En cuanto a su implementación, a ésta se le ha incluido un método de entrada por ratón robusto, la posibilidad de cambiar los valores de configuración del algoritmo y la posibilidad de mostrar resultados intermedios.

En cuanto a la configuración del algoritmo, los únicos parámetros configurables son los que definen el área de búsqueda en el que será buscado el objeto. Éstos se pueden configurar en horizontal y vertical simultáneamente o independientemente Además, se podrá indicar que la búsqueda se realice de forma exhaustiva.

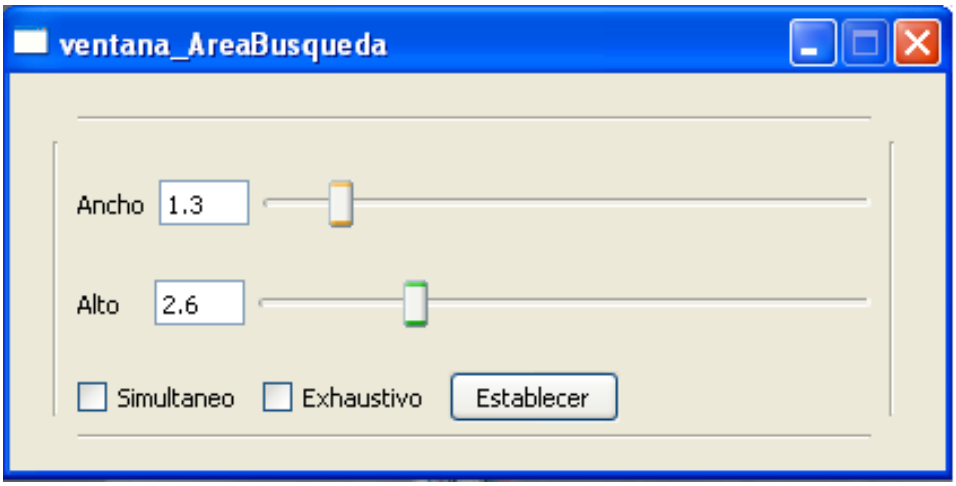

Figura 4-3: Ventana de ajuste del tamaño del área de búsqueda.

Debido a que la medida de confianza del algoritmo, basada en comparación de sumas absolutas, no ofrece una estimación precisa sobre si el objeto se ha localizado o no, se da la opción de mostrar los resultados intermedios en los que se podrá comparar visualmente el área de la imagen que el algoritmo estima como la localización del objeto y la plantilla seleccionada como modelo objeto. Para ello, el usuario tendrá que hacer click con el ratón en el botón "resultados".

Figura 4-4: Ventana resultados: estimación y plantilla (izquierda y derecha).

4.3 PKLT Fiter Tracker

Esta aplicación con interfaz gráfica tiene como objetivo la demostración y, a su vez, facilitación del uso de un algoritmo el cual ofrece muy buenos resultados en comparación con los *trackers* del estado del arte en la actualidad, el PKLTF.

Para la implementación de esta aplicación, se han seguido los pasos que se llevaron a cabo en el desarrollo de la aplicación Template Matching y, por lo tanto, se facilita una metodología para la integración y desarrollo de sus correspondientes interfaces gráficas de algoritmos de video-seguimiento de objetos en trabajos futuros.

La interfaz gráfica, a diferencia de la de Template Matching, ha sido desarrollada con programación visual, facilitando así la estructuración de los componentes de la ventana y dándole un mejor aspecto a la interfaz.

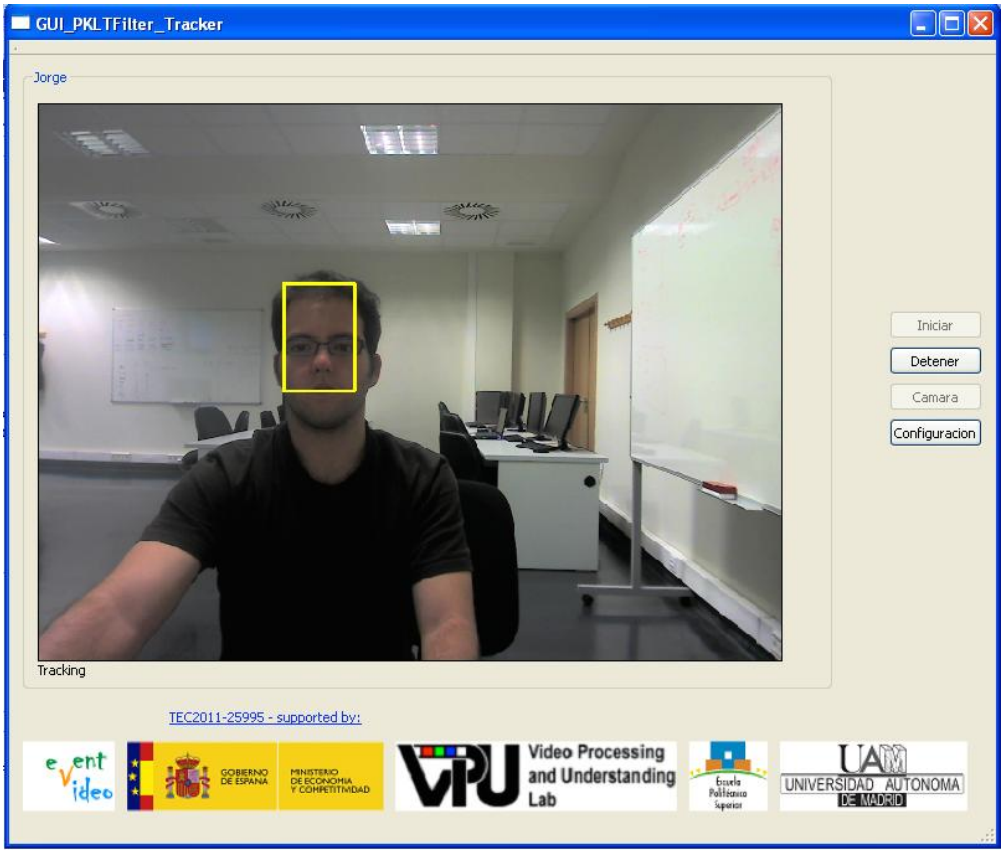

Figura 4-5: Selección del objeto en la aplicación PKLTF.

Debido a que el algoritmo dispone de una señalización en la que se estima si el algoritmo ha detectado el objeto o no, se ha implementado la salida con el *bounding box* de color verde si el objeto se ha localizado y en color rojo si el algoritmo estima que el objeto se ha perdido, en cuyo caso tratará de encontrarlo de manera más exigente.

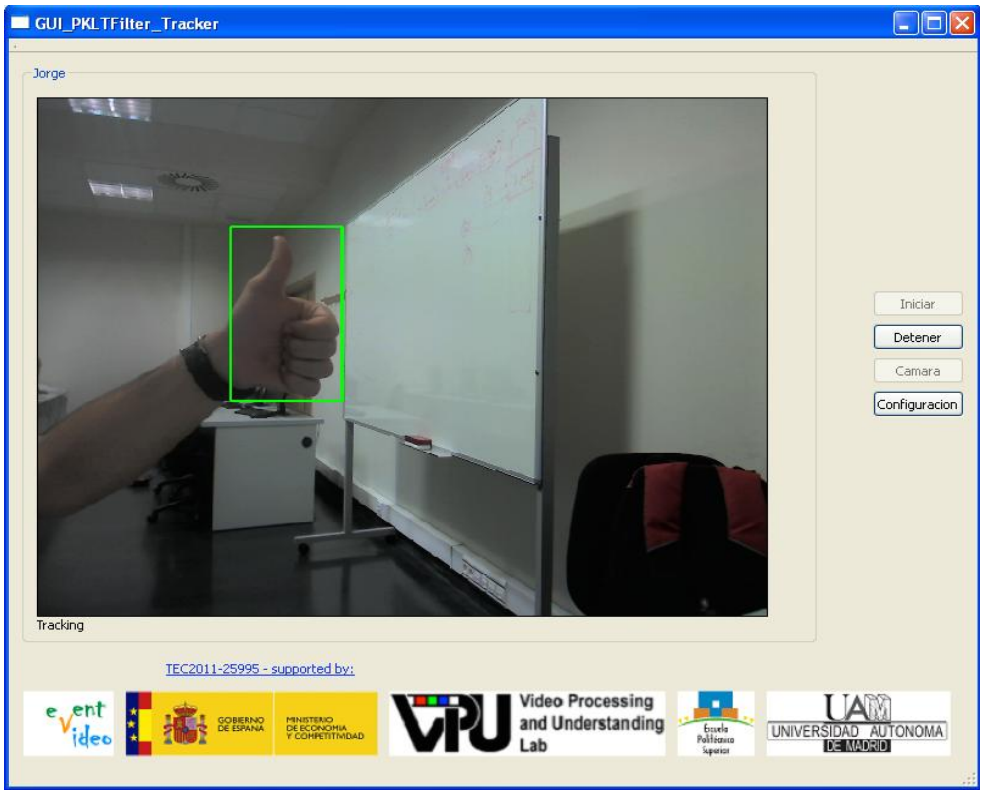

Figura 4-6: Objeto localizado.

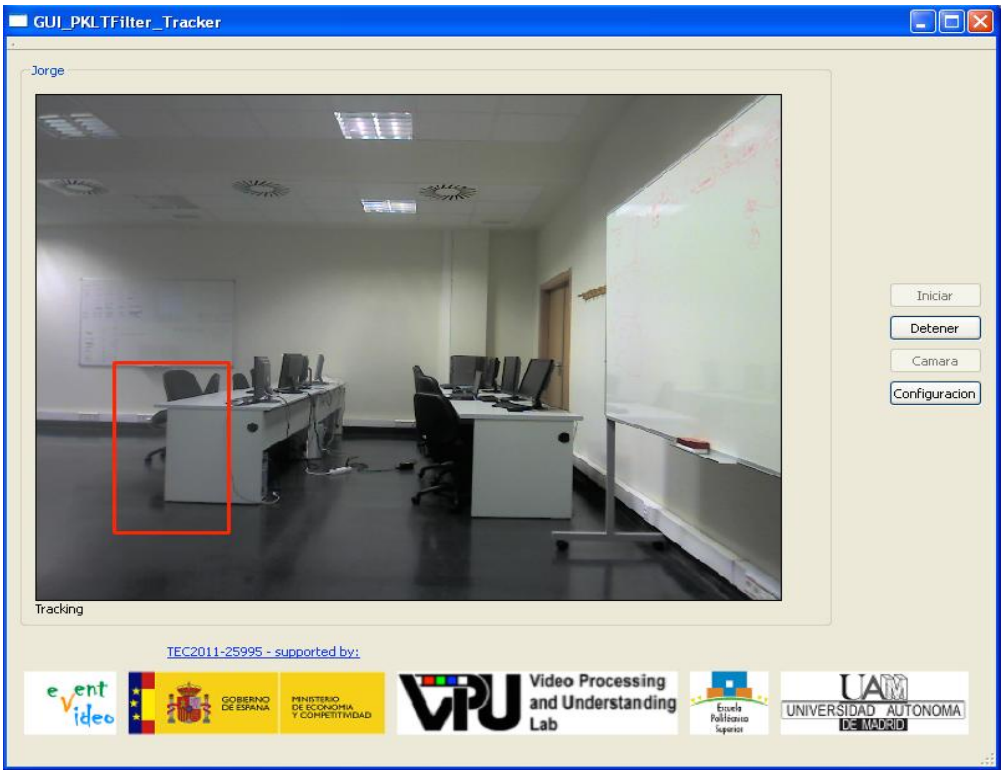

Figura 4-7: Objeto perdido.

En este nivel de integración (Nivel 3) se implementa la comunicación entre el usuario y el algoritmo de tal manera que, tanto la selección o re-selección del objeto o los parámetros del algoritmo se puedan cambiar en tiempo real. La primera será creada con el ratón y a la configuración se accederá haciendo click en el botón de "configuración".

En la ventana de configuración se podrá seleccionar si se desean visualizar las partículas con las que está trabajando el algoritmo, ajustar los parámetros de cada uno de los bloques del algoritmo o restablecer los valores por defecto existentes en el fichero de configuración del algoritmo.

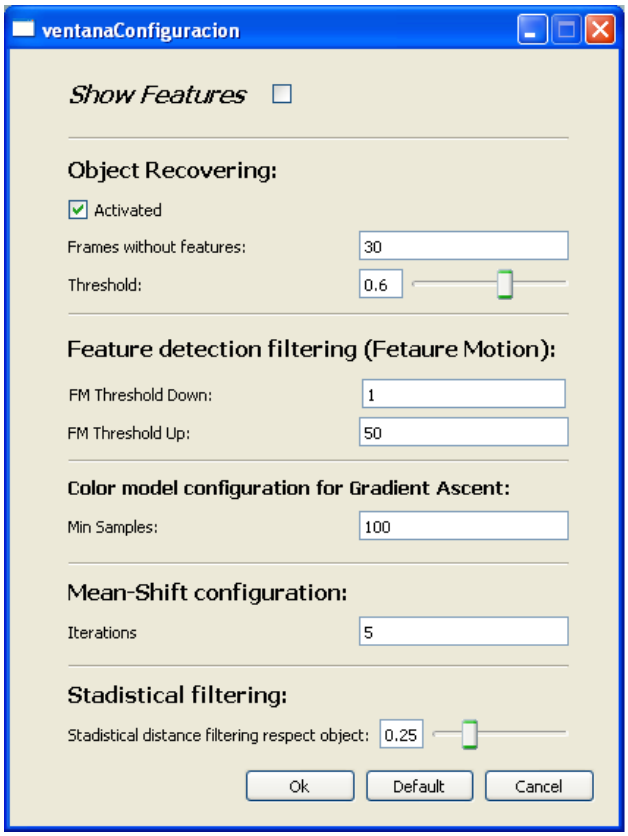

Figura 4-8: Parámetros de configuración PKLTF.

5 Marco de evaluación

Como en cualquier sistema de tratamiento digital de datos, en el video-seguimiento de objetos es muy importante hacer un estudio objetivo sobre el funcionamiento de la técnica utilizada. Por eso, en este trabajo, con motivo de verificar que los resultados de los algoritmos una vez insertados en la plataforma DiVA son coherentes con los esperados y verificar que, con las modificaciones propuestas se obtienen mejores resultados, se ha hecho uso de un *dataset*, sus correspondientes *ground-truth* y de un sistema de evaluación de seguidores de objetos.

5.1 Datasets

Un conjunto de datos de evaluación (*dataset)* para la evaluación de seguidores de objetos debe contar con una serie de secuencias de video en las que se contemplan los posibles problemas que pueden surgirle a un sistema de video-seguimiento de objetos de estas características, tales como: movimientos complejos, cambios de iluminación, ruido, oclusiones, etc. A su vez, un *ground-truth* de evaluación de video-seguimiento de objetos debe estar formado por datos supervisados por humanos sobre la localización del objeto a localizar en cada *frame* de cada secuencia de video del *dataset*.

A continuación se describen dos *datasets*. El primero se utilizó para el diseño y desarrollo de las mejoras implementadas (ver sección 6.2) y el segundo se utilizó para obtener resultados cuantitativos del sistema (ver sección 6.3), con el fin de comprobar el efecto de las mejoras implementadas.

5.1.1 Single Object Video Tracking dataset (SOVTds)

Con motivo de tener un *dataset* apropiado para la evaluación del video-seguimiento de objetos, el VPULab desarrolló este *dataset* en el que se combinan secuencias de video tanto sintéticas como reales con distintos problemas (descritos en 2.1), los cuales suponen un reto para los sistemas de video-seguimiento en la actualidad. Fue diseñado en [18] con el objetivo de tener una cantidad de secuencias suficientes como para realizar una comparación de *trackers* en todos sus aspectos y limitaciones.

Estas secuencias se dividen en cuatro niveles:

- **Nivel 1:** secuencias sintéticas. En ellas se hace un estudio controlado de cada uno de los problemas que pueden aparecer en los algoritmos de vídeo-seguimiento. En total se crearon 35 secuencias representando diversas situaciones con figuras geométricas.
- **Nivel 2:** secuencias de laboratorio simples. En total, 21 secuencias capturadas en un entorno controlado, pero real, en las que se plantean las diferentes dificultades a superar.
- **Nivel 3:** secuencias reales simples. Éstas recopilan un total de 53 secuencias extraídas de *datasets* ya existentes en los que el entorno en el que se capturaron no es controlado.

- **Nivel 4:** secuencias reales complejas. Éstas recopilan un total de 15 secuencias extraídas de distintos *datasets*, comprendiendo las situaciones más complejas en las que se pueden evaluar los algoritmos en los escenarios más realistas.

En total, 124 secuencias con las que es posible estudiar cada una de las limitaciones de los algoritmos de manera aislada (secuencias sintéticas o controladas) o en entornos realistas no controlados (niveles 3 y 4).

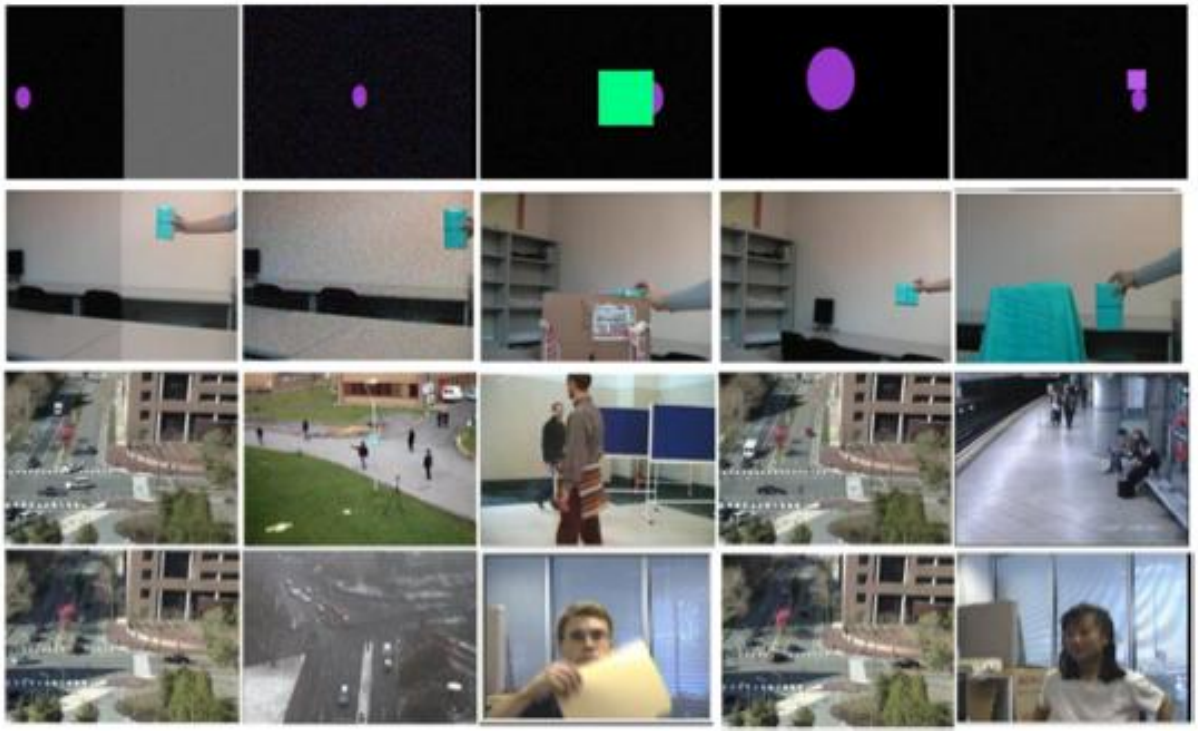

Figura 5-1: Muestras de los diferentes niveles del dataset SOVTds (una fila por nivel).

5.1.2 Visual Object Tracking challenge (VOT2013)

Los retos VOT (Video Object Tracking Challenge) [19] ofrecen a la comunidad de investigadores del video-seguimiento de objetos la oportunidad de comparar, sobre secuencias a corto plazo, los resultados obtenidos con diferentes algoritmos y/o implementaciones a nivel mundial. De esta manera además, se puede determinar y discutir cómo han evolucionado las técnicas en el ámbito del video-seguimiento de objetos. A estos retos, los miembros de la comunidad pueden apuntarse y así comparar sus resultados con los de todo el mundo y así aportar en la evolución de estos sistemas de video-procesado.

El reto VOT utilizado en este trabajo es el VOT2013. Este reto tiene como objetivo la comparación de seguidores de objetos mono-objetivo en escenas realistas en las que se producen típicas situaciones a las que se enfrentan estos sistemas. De esta manera, el reto proporciona dos paquetes descargables en la red:

- Sistema software de evaluación: en Matlab y Octave.
- Dataset: repositorio de vídeos y *ground-truth*.

El *dataset* consta con un total de 16 vídeos elegidos para evaluar ciertos problemas específicos con diferentes objetos como se puede observar en la siguiente tabla:

Name	number of frames	Description
bicycle	271	bike, occlusion, scale change
bolt	350	body, articulation, scale change
car	374	car
cup	303	cluttered background
david	770	head, ilumination and scale change
diving	231	body, articulated, rotation
face	415	head, occlusion
gymnastics	207	body, articulated, scale change
hand	244	hand, articulated
iceskater	500	body, articulated
juice	404	box
jump	228	bike, scale change
singer	351	body,ilumination and scale change
sunshade	172	head, ilumination change
torus	264	interesting geometry
woman	597	body, occlusion, scale change

Tabla 5-1 : Descripción del repositorio de vídeos en VOT2013.

En cuanto al *groun-truth*, que define la posición del objeto en cada *frame* de cada vídeo, se determina con la consideración de que es muy complicado determinar la posición del objeto de forma objetiva. Esta localización es seleccionada manualmente por los autores del repositorio.

5.2 Métricas de evaluación

Para la evaluación de algoritmos de video-seguimiento de objetos, además de ser necesario un repositorio de secuencias de vídeo y sus ficheros de *groun truth* correspondientes, se debe utilizar un sistema de medida de los resultados que dé una estimación lo más acertada y objetiva sobre su funcionamiento.

En la actualidad, según se estudia en [20], existen diversas medidas para la estimación de aciertos del algoritmo. Cada autor usa sus propias métricas y, además, se utilizan repositorios de videos con escasas secuencias de videos. Por lo tanto, la tarea de comparación de algoritmos es complicada ya que no está estandarizado entre el colectivo de investigadores en materia de vídeo-seguimiento de objetos.

En cuanto a las métricas, en [20] se hace un estudio de la correlación existente entre distintos tipos de medidas para así hallar la redundancia que aparece entre distintos tipos de medidas. En la siguiente tabla se muestran los resultados de correlación del estudio. Las métricas tienen un rango de 0 a 1 en el que, cuanto mayor sea el valor, mejores son los resultados del algoritmo. Para la estimación de estas medidas, se hizo uso del *dataset* SOVTds descrito en 5.1.1:

	SFDA	ATA	ATEi	AUCi	Overlap	CTM	TC	CoTPSi
SFDA	1,00	0,99	0,64	0,96	0,96	0,99	0,89	0,88
ATA	0,99	1,00	0,64	0,96	0,96	0,99	0,89	0,88
ATEi	0,64	0,64	1,00	0,60	0,60	0,64	0,61	0,42
AUCi	0,96	0,96	0,60	1,00	0,99	0,96	0,89	0,92
Overlap	0,96	0,96	0,60	0,99	1,00	0,96	0,89	0,92
CTM	0,99	0,99	0,64	0,96	0,96	1,00	0,89	0,88
TC	0,89	0,89	0,61	0,89	0,89	0,89	1,00	0,78
CoTPSi	0,88	0,88	0,42	0,92	0,92	0,88	0,78	1,00

Tabla 5-2 : Correlación entre diferentes medidas de evaluación.

Según los resultados, la mayoría de las medidas descritas tienen una alta correlación entre ellas (en torno al 90%) y, por lo tanto, se obtienen buenos resultados independientemente de la métrica escogida.

La métrica con la que trabaja el software de comparación de *trackers* del VPULab, utilizado para este trabajo (ver sección 6.3), es SFDA (Sequence Frame Detection Acuracy). Ésta calcula, para cada *frame*, el solapamiento espacial entre la estimación de la localización del objeto y la anotación existente en el *ground truth* y, además, contiene información sobre el número de objetos detectados, pérdidas detectadas y falsos positivos.

La métrica queda definida como:

$$SFDA = \frac{\sum_{t=1}^{t=N\,frames} FDA(t)}{\sum_{t=1}^{t=N\,frames} \exists\left(N_G^{(t)} OR\, N_D^{(t)}\right)}$$

$$FDA(t) = \frac{overlap\_ratio}{\left[\left(N_G^{(t)} + \ N_D^{(t)}/2\right)\right]}$$

$$overlap\_ratio = \sum_{i=1}^{N_{mapped}^{(t)}} \frac{\left|G_i^{(t)} \cap D_i^{(t)}\right|}{\left|G_i^{(t)} \cup D_i^{(t)}\right|}$$

Dónde:

- $G_i^{(t)}$ es la anotación de *ground-truth* correspondiente al *frame* t.
- $D_i^{(t)}$ es la anotación de la estimación de la localización correspondiente al *frame* t.
- $N_G^{(t)}$ y $N_D^{(t)}$ corresponden el número de anotaciones de *ground-truth* y los objetos localizados en el *frame* t respectivamente.
- $N\ frames$ es el número de *frames* de la secuencia.
- $N_{mapped}^{(t)}$ es el número de asociaciones entre *ground-truth* y objetos localizados en el *frame* t.

6 Mejoras del algoritmo PKLT

En este apartado se describen las mejoras realizadas en el algoritmo PKLT Filter Tracker, originalmente implementado por Antonio González Huete. Para esto, ha sido necesario realizar un estudio completo sobre el algoritmo tanto a nivel teórico como a nivel de implementación.

Las dos mejoras realizadas se han centrado en la problemática de crear un modelo de objeto en el que el fondo entra dentro del área del *bounding box* y estudiar la relación coste computacional/resultados en la actualización del modelo.

6.1 Limitaciones del algoritmo

A pesar de los buenos resultados del algoritmo, éste no realiza una estimación exacta sobre la localización del objeto en todo momento. Se puede decir que la combinación de características, tanto de vectores de movimiento como de color, hacen que este algoritmo sea bastante robusto ante los problemas típicos de los *trackers*, descritos en la sección 2.1.

Realizando experimentos sobre el algoritmo con secuencias de video de repositorios en *datasets* y con secuencias de video captadas en tiempo real, se puede considerar este algoritmo robusto a cambios de escala, rotaciones graduales, movimientos bruscos o complejos, cambios de iluminación graduales y, en general, si se pierde el objeto, es capaz de recuperarlo con bastante rapidez y certeza. Sin embargo, el algoritmo es menos robusto ante la aparición de objetos similares en la escena, oclusiones parciales o totales y rotaciones complejas.

6.1.1 Fondo

El algoritmo PKLT Filter Tracker, como se describe en la sección 2.2.9, hace uso de una etapa de corrección de color basado en Mean-Shift y, al ser éste un algoritmo basado en información de color, tiene la desventaja de que, cuando el color de alguna parte del fondo es similar a alguno de los colores característicos que definen el objeto, se pueden producir errores en la localización del objeto.

Con motivo de reducir este tipo de errores, se puede hacer una corrección del histograma a partir del cual se creará el modelo de objeto. Esta técnica consiste en estimar el histograma del fondo para así poder reducir el peso de la información de fondo que está presente en el histograma que define el objeto.

6.1.2 Problemas de actualizar el modelo

El algoritmo PKLT Filter Tracker realiza una actualización del modelo con el que el objeto queda definido. Por lo tanto, el objeto seguido no es exactamente el que se define en la inicialización si no que, en cada iteración se realiza una nueva estimación del modelo a seguir con el criterio de que se adapte lo máximo posible con respecto a los cambios que el

objeto ha ido adquiriendo a lo largo de la secuencia. Esta actualización se realiza con motivo de dotar al algoritmo con capacidad para funcionar a largo plazo.

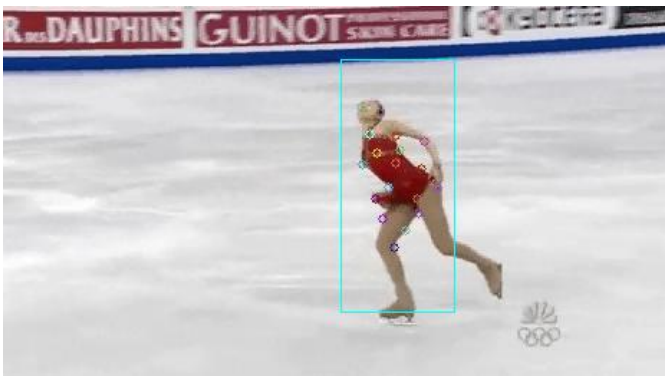

Figura 6-1: puntos característicos en un *frame*.

Figura 6-2: puntos característicos actualizados en un *frame* posterior.

En las dos figuras anteriores, correspondientes a la salida del algoritmo para la secuencia "iceskater" del repositorio disponible en VOT2013, se muestran los puntos característicos que actualizan, en el segundo *frame*, el modelo del objeto a seguir. Estos puntos característicos corresponden, en su mayoría, a los bordes del objeto. Por lo tanto, debido a que, tras procesar cada frame, el algoritmo actualiza el histograma extrayendo información alrededor de los puntos característicos, se introduce información de color del fondo (no deseada). Se deduce por lo tanto que, si bien el algoritmo es capaz de funcionar a largo plazo, el modelo de objeto se irá distorsionando a lo largo de la secuencia.

6.2 Experimentos

Tras estudiar el algoritmo y su implementación en profundidad, se analizaron las posibles mejoras que podrían ser realizadas. Estas mejoras se hicieron de tal manera que no se modificase el funcionamiento original del algoritmo.

Para arreglar la problemática que aparece en la etapa de ajuste en la localización del algoritmo (basado en color), se ha realizado una corrección del histograma de color que

modela el objeto para así eliminar la información de color del fondo que pueda aparecer en él. La técnica utilizada está basada en la que se utiliza en el algoritmo CBWH (descrito en la sección 2.2.7).

La idea básica en esta corrección es la de estimar qué colores del fondo aparecen en el *bounding box* que selecciona el objeto para así estimar la corrección que se debe aplicar al histograma calculado con los píxeles del interior del *bounding box*. Para ello, una vez seleccionado el cuadrilátero que define el área en el que se encuentra el objeto, se calcula su histograma y, además, con motivo de obtener información de los colores del fondo que pueden haberse introducido en la definición original del objeto, se realiza una expansión del área que define al objeto al doble de su tamaño.

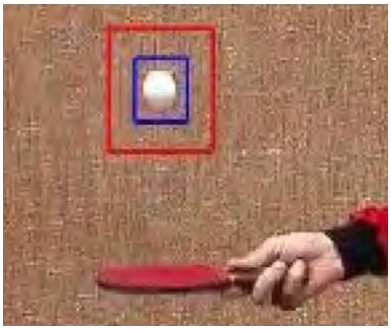

Figura 6-3: Definición del objeto y expansión de su área. (Azul y Rojo Respectivamente) [8].

En la nueva zona considerada entre el nuevo área y la selección original del objeto, se calcula un nuevo histograma con el que se extrae la información de color sobre el fondo. Esta información es utilizada para crear una máscara con la cual se reduce el peso de aquellos valores del histograma con el que se modela el objeto que corresponden al fondo. Por último, se reajustan los pesos de los valores relevantes del histograma para no perder información en procesos de normalización que aparecen en la implementación.

En cuanto a la limitación en la actualización de fondo, se implementa en la configuración del algoritmo desde fichero la opción de actualizar o no el modelo en cada *frame.*

6.3 Evaluación resultados

Tras experimentar con el algoritmo centrándose sus dos limitaciones anteriormente descritas, se evalúan los resultados obtenidos para así tener un estudio objetivo sobre sus mejoras. Para esta evaluación, se han realizado dos estudios; uno de resultados objetivos de búsqueda basado *ground truth* y un estudio de tiempos de ejecución. Todos los estudios tienen en cuenta la implementación ya existente y cada una de las mejoras realizadas.

6.3.1 Estudio de resultados basado en ground truth

Para la evaluación objetiva de los resultados se ha utilizado un software de evaluación de *trackers* disponible en el VPULab.

Para la evaluación de resultados con este software, es necesario que el algoritmo tenga como salida un fichero de texto debidamente estructurado en el que, para cada *frame* de la secuencia de video, se obtengan el número de *frame* y las cuatro coordenadas que definen el cuadrilátero que encuadra el resultado de la localización del objeto en ese instante. Por lo tanto, se ha creado un sistema en el que, indicándole un directorio en el que se encuentran los vídeos para la evaluación y un directorio que contiene los *ground truth* a partir de los cuales se inicializa la búsqueda, éste genera automáticamente los datos de salida para su evaluación, facilitando un sistema de salida de resultados de *trackers* para futuros trabajos.

En esta evaluación se han comparado los algoritmos descritos en el estado del arte de este trabajo, los cuales comprenden los algoritmos clásicos para el seguimiento de objetos y algunos de los algoritmos que a día de hoy dan mejores resultados y, además, el punto de partida del algoritmo PKLT Filter Tracker y las mejoras que se la han realizado sobre él en este trabajo.

A continuación se muestran una serie de figuras en las que se muestran los resultados de salida de la comparación para los 16 vídeos del repositorio del reto VOT2013 y los algoritmos descritos en el estado del arte de este trabajo. La métrica con la que se han comparado es SFDA, descrita en la sección 5.2.

Nota: Las correspondencias en la leyenda son:

- **Las ocho primeras:** corresponden a los algoritmos descritos en el estado del arte (mismo orden)
- **PKLTF:** Corresponde al punto de partida en el que se miden los resultados de PKLTF.
- **PKLTFnU:** Corresponde a la no actualización del modelo de objeto (*no Update*).
- **PKLTF-CB:** Corresponde a la mejora en la que se corrige el histograma del fondo.
- **PKLTFnU-CB:** Corresponde a las dos mejoras juntas; corrección de histograma y no actualización del modelo.

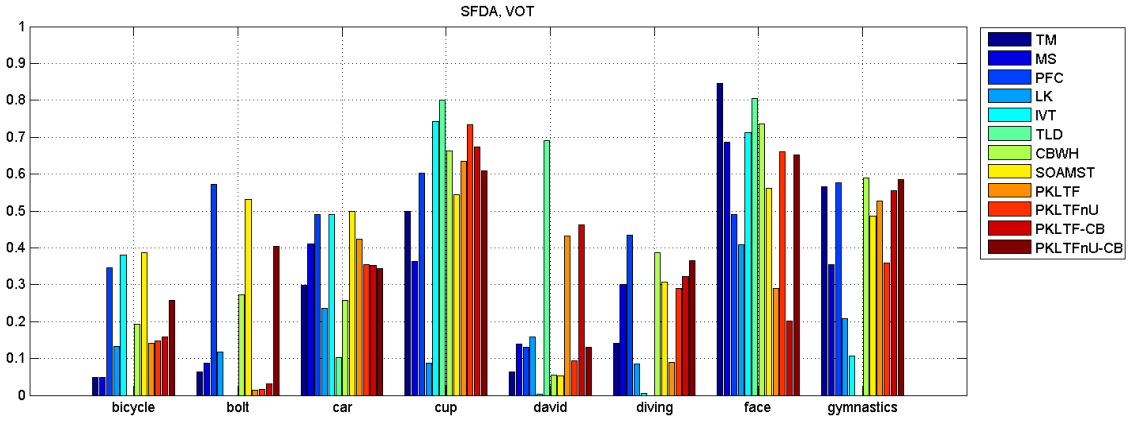

Figura 6-4: resultados del comparador, vídeos 1 al 8 de VOT2013.

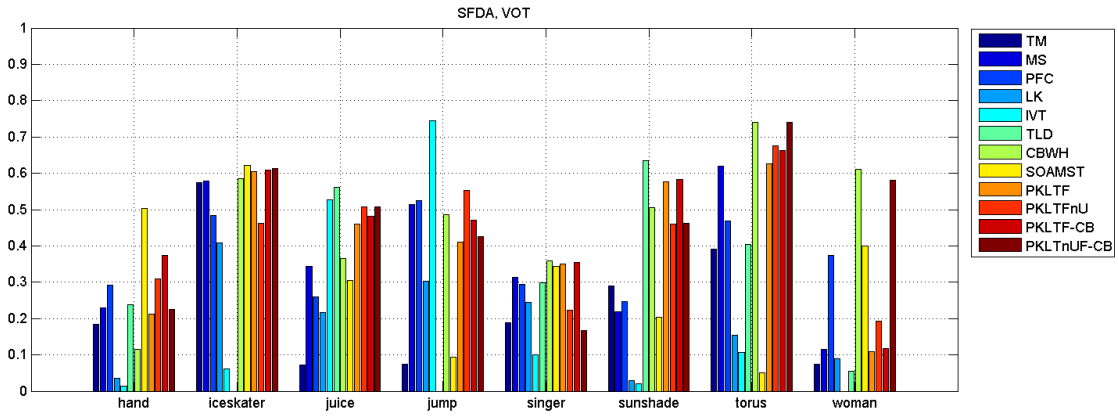

Figura 6-5: resultados del comparador, vídeos 9 al 16 de VOT2013.

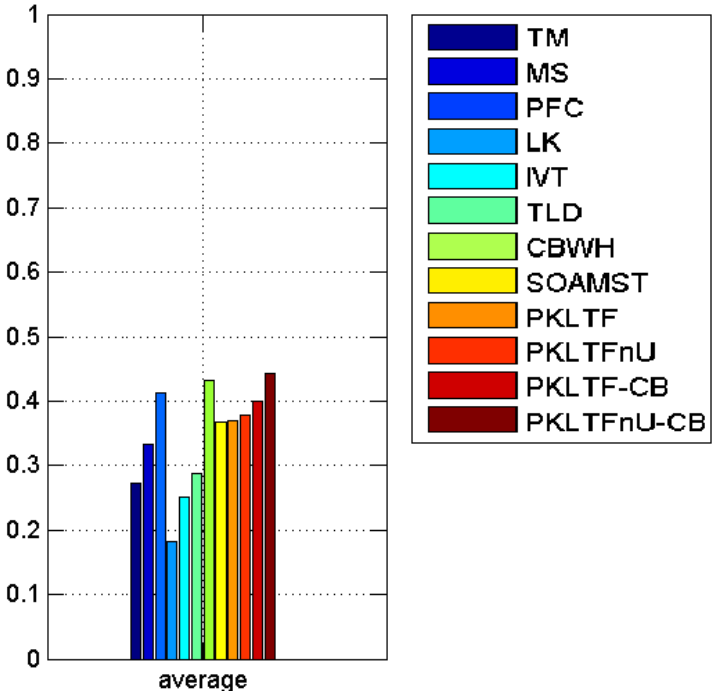

Figura 6-6: Media del resultado del comparador para los 16 vídeos de VOT2013.

Algoritmo / Video	TM	MS	PFC	LK	IVT	TLD	CBWH	SOAMST	PKLT	PKLTnU	PKLT-CB	PKLTFnU-CB
bicycle	0,048	0,048	0,345	0,132	0,381	0,000	0,192	0,386	0,142	0,147	0,158	0,258
bolt	0,063	0,087	0,572	0,118	0,000	0,000	0,272	0,531	0,015	0,017	0,031	0,403
car	0,298	0,411	0,490	0,237	0,491	0,102	0,258	0,500	0,423	0,355	0,351	0,344
cup	0,498	0,363	0,602	0,087	0,742	0,800	0,664	0,544	0,635	0,735	0,673	0,610
david	0,064	0,139	0,131	0,158	0,004	0,691	0,056	0,052	0,433	0,094	0,462	0,131
diving	0,140	0,301	0,435	0,086	0,006	0,000	0,386	0,308	0,089	0,289	0,323	0,365
face	0,846	0,687	0,490	0,409	0,713	0,805	0,736	0,560	0,291	0,661	0,201	0,652
gymnastics	0,567	0,354	0,577	0,208	0,108	0,000	0,590	0,486	0,528	0,359	0,555	0,585
hand	0,184	0,230	0,291	0,036	0,015	0,238	0,114	0,504	0,212	0,310	0,374	0,226
iceskater	0,574	0,579	0,485	0,409	0,061	0,000	0,585	0,622	0,605	0,461	0,609	0,613
juice	0,073	0,343	0,260	0,218	0,528	0,561	0,366	0,305	0,460	0,509	0,481	0,508
jump	0,075	0,514	0,524	0,304	0,744	0,000	0,487	0,094	0,411	0,553	0,471	0,426
singer	0,188	0,314	0,295	0,244	0,100	0,299	0,359	0,344	0,350	0,224	0,354	0,167
sunshade	0,290	0,219	0,247	0,029	0,020	0,634	0,505	0,204	0,576	0,460	0,582	0,462
torus	0,391	0,619	0,469	0,155	0,107	0,404	0,740	0,050	0,626	0,675	0,663	0,739
woman	0,075	0,115	0,375	0,090	0,000	0,056	0,612	0,399	0,110	0,193	0,117	0,581
Media	0,273	0,333	0,412	0,183	0,251	0,287	0,433	0,368	0,369	0,378	0,400	0,442

Tabla 6-1 : Valores de salida del comparador SFDA con VOT2013 (valores de las 3 figuras anteriores).

Como se puede observar, verificando que con las modificaciones realizadas se obtienen mejores resultados, se demuestra que, para cada una de las mejoras realizadas se han obtenido mejores resultados, en promedio, que los que se obtenían en el punto de partida (PKLTF). Sin embargo, ejecutando el algoritmo al tiempo que se representaban los resultados, se observaba que, para algunas secuencias de video, la representación de los resultados no era a tiempo real aunque los resultados objetivos fuesen mejores.

En cuanto a la no actualización del modelo, se deduce que se obtienen mejores resultados siempre que no haya cambios importantes en la iluminación (como en la secuencia "David") o cambios de escala notables (secuencia "singer"). En secuencias en las que el objeto no cambia su apariencia de manera significativa (secuencas "woman" o "bolt"), la no actualización del modelo en combinación con un modelo de objeto no influenciado por la distorsión del fondo, hace que este algoritmo proporcione unos resultados notablemente mejores tras la implementación de las mejoras. Por lo tanto, se puede deducir que la combinación de ambas ofrece mejores resultados que cada una de ellas por separado.

Además, este algoritmo cuenta con parámetros de configuración con los que se podría ajustar el funcionamiento a las características de la escena, pero, con el objetivo de obtener un estudio más objetivo se han realizado los experimentos con los valores por defecto para todas las secuencias. Si estos parámetros se hubiesen ajustado de manera adecuada, es posible que los resultados hubiesen mejorado.

6.3.2 Estudio de resultados en los tiempos de ejecución

Para obtener un estudio más exhaustivo y, como indica el título de este trabajo, estudiar el funcionamiento del algoritmo en tiempo real, se han realizado medidas sobre los tiempos de ejecución del algoritmo para cada uno de los casos estudiados; punto de partida y las dos modificaciones. Además, se han comparado los tiempos de ejecución con el bloque de representación de resultados activados y desactivados. De esta manera, podremos saber el tiempo de procesado independientemente del uso de recursos que la representación consume.

Podremos decir que el sistema funciona a tiempo real siempre y cuando el tiempo de procesado no sea mayor al tiempo de recepción de los *frames* de cada secuencia de vídeo. Además, al tratarse de secuencias de vídeo, estas deben tener un *frame rate* de en torno a 25 fps (*frames per second)* mínimo en la representación para que el sistema visual humano lo perciba con continuidad.

Con estas consideraciones, a continuación se muestran los resultados obtenidos en la medida de los tiempos de procesado. En ellos se incluye la media para poder tener una visión general de los resultados. Sin embargo, en función de la secuencia de video, los resultados pueden variar notablemente.

Video	PKLTF	PKLTFnoUpdate	PKLTF&CBWH	PKLTF&CBWHnoUpdate
bicycle	20,8	20,8	18,1	20,8
bolt	13,0	10,0	7,0	17,5
car	34,0	41,6	62,3	41,6
cup	37,9	33,7	33,7	12,1
david	32,1	22,6	32,1	25,7
diving	38,5	46,2	46,2	38,5
face	34,6	34,6	31,9	20,8
gymnastics	34,5	41,4	34,5	20,7
hand	32,0	34,9	40,7	32,0
iceskater	22,7	17,2	21,7	22,7
juice	33,7	31,1	33,7	33,7
jump	20,7	22,8	20,7	2,2
singer	16,7	10,6	16,7	9,8
sunshade	34,4	43,5	34,4	34,4
torus	37,7	29,3	37,7	29,3
woman	26,0	21,3	31,4	27,1
Media	29,3	28,9	31,4	24,3

Tabla 6-2 : Tiempos de procesado con representación por pantalla (medidas en *frames* por segundo).

Video	PKLTF	PKLTFnoUpdate	PKLTF&CBWH	PKLTF&CBWHnoUpdate
bicycle	24,6	27,1	24,6	30,1
bolt	14,0	11,3	7,4	20,6
car	74,8	74,8	74,8	62,3
cup	60,6	60,6	60,6	13,8
david	48,1	29,6	48,1	35,0
diving	77,0	77,0	77,0	77,0
face	41,5	37,7	37,7	24,4
gymnastics	69,0	69,0	51,8	28,9
hand	81,3	61,0	61,0	61,0
iceskater	29,4	21,7	31,3	29,4
juice	67,3	57,7	57,7	67,3
jump	25,3	25,3	25,3	2,2
singer	20,6	12,1	20,6	10,3
sunshade	57,3	86,0	57,3	57,3
torus	66,0	52,8	66,0	52,8
woman	39,8	26,0	49,8	35,1
Media	49,8	45,6	46,9	38,0

Tabla 6-3 : Tiempos de procesado sin representación por pantalla (medidas en *frames* por segundo).

A la vista de los resultados en la evaluación de tiempos de cómputo, se puede deducir que, para el caso en el que se muestran los resultados por pantalla, es un sistema capaz de localizar objetos en tiempo real ya que su media de tiempos para los 16 vídeos estudiados proporciona un *frame rate* superior al necesario para percibir la secuencia de manera continua. Por otro lado, sin la representación, el sistema ofrece unos tiempos de cómputo con bastante margen suponiendo que las secuencias de vídeo no suelen superar los 25 *frames* por segundo. Se puede observar que, para algunas secuencias en concreto (por

ejemplo, “jump” y “singer”), los tiempos de cómputo son notablemente superiores que para el resto de las secuencias en los experimentos realizados para la corrección de fondo y no actualización del modelo (última columna en la tabla). Esto se debe a que, si no se realiza la actualización del modelo y, tras procesar el algoritmo para reducir la distorsión introducida por el fondo, la distancia entre histogramas es mayor, considerando el algoritmo que el objeto se ha perdido pese a que se está siguiendo correctamente e iniciando una búsqueda más exhaustiva con motivo de recuperar el objeto, obteniendo así unos tiempos de cómputo mayores a pesar de los mejores resultados obtenidos. Para evitar este comportamiento se debería aumentar el umbral mínimo de similitud entre histogramas que define cuando un objeto se ha perdido. Este umbral no se ha modificado ya que implicaría un estudio en mayor profundidad de las modificaciones de todos los parámetros del algoritmo y esto no entra dentro de los objetivos del sistema, ya que además se considera más interesante la futura implementación de otro método de actualización del modelo (u otras mejoras que se comentan en trabajo futuro).

7 Conclusiones y trabajo futuro

7.1 Conclusiones

En este trabajo se define la metodología para la integración de algoritmos de vídeo-seguimiento de objetos en una plataforma de vídeo vigilancia distribuida concreta, DiVA. Además, también se establece una posible metodología para el desarrollo de interfaces gráficas que sirvan de demostradores de este tipo de algoritmos. Con estas aportaciones, en futuros trabajos, existirá al menos una manera de realizar el trabajo que se ha llevado a cabo en éste, facilitando esta tarea y minimizando los tiempos de aprendizaje que lleva la adaptación a este entorno de desarrollo.

Por otro lado, se ha mejorado el algoritmo PKLTF teniendo en cuenta el fondo del objeto a seguir para así no distorsionar su modelo asociado. Además, haciendo un estudio sobre la implementación del algoritmo en la cual se provee al algoritmo de auto-adaptación para funcionar a largo plazo, se ha observado que la forma en la que trabaja el algoritmo distorsiona el modelo de objeto introduciendo nuevamente componentes de color del fondo. Con esto, se ha mejorado el algoritmo original y se ha hecho un estudio objetivo sobre él.

Parte de este trabajo ha sido presentado en el Workshop: "Strategies for object Segmentation, Detection and Tracking in Complex Enviroments for Event Detection in Video Surviellance and Monitoring", TEC2011-25995 EventVideo (2012-2014).

7.2 Trabajo futuro

El trabajo a realizar en materia de vídeo-seguimiento de objetos es inmenso y, a pesar de que a día de hoy existen aplicaciones que ofrecen muy buenos resultados, existen múltiples ramas de investigación para desarrollar y mejorar este tipo de algoritmos: aún hay múltiples aspectos en los que se tiene que seguir trabajando como son unos buenos métodos de evaluación estandarizados, robustez ante los diferentes problemas que aparecen en el vídeo-seguimiento, aplicación de esta tecnología en sistemas de usuario, etc. Además, siguiendo la línea de este trabajo, se puede trabajar en la integración de nuevos algoritmos desarrollados o en desarrollo con el fin de mejorarlos y poder realizar pruebas con la interfaz de usuario.

Respecto al algoritmo estudiado, PKLTF, la actualización del modelo de objeto le da una gran robustez en cuanto al funcionamiento a largo plazo. Sin embrago, la propia actualización genera modelos de objeto actualizados, los cuales se distorsionan con información del fondo y, por lo tanto, podría trabajarse en la realización de un modelo objeto iterativamente en el que no se introdujese información del fondo.

Como mejoras de PKLTF, además se puede trabajar en:

- Creación de un modelo por partes: proporcionaría robustez a oclusiones parciales y/o rotaciones.
- Añadir información de forma al objeto: el algoritmo fue diseñado para combinar distintos tipos de características. En la actualidad combina: movimiento y color.

- Implementación de un método que permita aislar objetos similares.
- Diseño de un método de actualización del modelo de objeto en el que no se distorsione el modelo con el color del fondo, generando robustez ante cambios de apariencia del objeto y, por lo tanto, proveerle de una buena funcionalidad a largo plazo.
- Combinar el sistema con un bloque de detección y aprendizaje, aplicando ideas similares a las que se usan en el *tracker* TLD (sección 2.2.6).
- Optimizar el proceso de recuperación del objeto si éste se considera perdido. Es un proceso muy costoso que penaliza los tiempos de cómputo del algoritmo.
- Adaptación automática de los parámetros en función de las características de la escena (cantidad de movimiento, iluminación, contraste, etc).

Referencias

[1] E. Maggio and A. Cavallaro, Video Tracking: Theory and Practice. Wiley, 2011.
[2] R. Brunelli, Template Matching Techniques in Computer Vision: Theory and Practice. Wiley Publishing, 2009.
[3] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking", In trans. On Pattern Analysisand Machine Intelligence, vol. 25, no. 5, pp. 564-577, 2003.
[4] K. Nummiaro, E. Koller-Meier, and L. V. Gool, "An adaptive colour-based particle flter," In proc. of Image and Vision Computing, vol. 21, no. 1, pp. 99-110, 2002.
[5] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," International Journal of Computer Vision, vol. 56, pp. 221-255, March 2004.
[6] D. A. Ross, J. Lim, R. S. Lin, and M. H. Yang, "Incremental learning for robust visual tracking," International Journal of Computer Vision, vol. 77, pp. 125-141, May 2008.
[7] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," In trans. On Pattern Analysis and Machine Intelligence, vol. 34, no. 7, pp. 1409-1422, 2011.
[8] J. Ning, L. Zhang, D. Zhang, and C. Wu, "Robust mean-shift tracking with corrected background-weighted histogram," Computer Vision, IET, vol. 6, no. 1, pp. 62-69, 2012.
[9] J. Ning, L. Zhang, D. Zhang, and C. Wu, "Scale and orientation adaptive mean shift tracking," Computer Vision, IET, vol. 6, no. 1, pp. 52-61, 2012.
[10] A. González, "Seguimiento y producción automática mediante cámaras PTZ en entornos de red," Master's thesis. Universidad Autónoma de Madrid 2013.
[11] J. Shi and C. Tomasi, "Good Features to Track," In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 593-600, June 1994.
[12] C. Harris and M. Stephens "A Combined Corner and Edge Detector," In Procceding Alvey Vision Conference, Univ. Manchester, pp. 147-151, 1988.
[13] D. Comaniciu, V. Ramesh, P. Meer: Real-time tracking of non-rigid objects using mean shift. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00), Vol. 2, pp 142–149, 2000.
[14] San Miguel, J. C., Bescós, J., Martínez, J. M., & García, Á. (2008, May). DiVA: a Distributed Video Analysis framework applied to video-surveillance systems. In Image Analysis for Multimedia Interactive Services, 2008. WIAMIS'08. Ninth International Workshop on (pp. 207-210). IEEE.
[15] C. Sánchez, "Entorno de desarrollo de aplicaciones de video-seguridad multicámara," Master's thesis, Universidad Autónoma de Madrid, 2014.
[16] QT Project (http://qt-project.org/)
[17] OpenTLD (http://www.gnebehay.com/tld/)
[18] M. Lozano, "Evaluación comparativa de algoritmos de seguimiento de objetos (tracking)," Master's thesis, Universidad Autónoma de Madrid, 2012.
[19] VOT2013(http://www.votchallenge.net/vot2013/)
[20] R. Martín and J.M. Martínez, "Correlation study of video object trackers evaluation metrics," The Institution of Engineering and Technology, Electronic Letters, vol 50, Issue 5, pp. 361-363, 2014